The main pillars of learning programming — and why beginners should master them.

Translation: 学习编程的主要支柱——以及为什么初学者应该掌握它们。

I have been programming for more than 20 years. During that time, I've had the pleasure to work with many people, from whom I learned a lot. I've also worked with many students, coming fresh from university, with whom I had to take on the role of a teacher or mentor.

Translation: 我已经编程20多年了。在那段时间里，我很高兴能和很多人一起工作，从他们那里我学到了很多。我还与许多刚从大学毕业的学生合作过，我不得不与他们一起扮演老师或导师的角色。

Lately, I have been involved as a trainer in a program that teaches coding to absolute beginners.

Translation: 最近，我作为一名培训师参与了一个向绝对初学者教授编码的项目。

Learning how to program is hard. I often find that university courses and bootcamps miss important aspects of programming and take poor approaches to teaching rookies.

Translation: 学习如何编程很难。我经常发现大学课程和训练营错过了编程的重要方面，对新手的教学方法也很差。

I want to share the five basic pillars I believe a successful programming course should build upon. As always, I am addressing the context of mainstream web applications.

Translation: 我想分享我认为一门成功的编程课程应该建立在五个基本支柱之上。和往常一样，我正在处理主流web应用程序的上下文。

A rookie's goal is to master the fundamentals of programming and to understand the importance of libraries and frameworks.

Translation: 新手的目标是掌握编程的基本原理，并理解库和框架的重要性。

Advanced topics such as the cloud, operations in general, or build tools should not be part of the curriculum. I am also skeptical when it comes to Design Patterns. They presume experience that beginners never have.

Translation: 高级主题，如云、一般操作或构建工具，不应成为课程的一部分。当谈到设计模式时，我也持怀疑态度。他们假定初学者从未有过的经验。

So let’s look at where new programmers should start.

Translation: 所以，让我们看看新程序员应该从哪里开始。

Test-Driven Development (TDD)

Translation: 测试驱动开发（TDD）

TDD brings a lot of benefits. Unfortunately, it is an advanced topic that beginners are not entirely ready for.

Translation: TDD带来了很多好处。不幸的是，这是一个初学者还没有完全准备好的高级话题。

Beginners shouldn’t write tests. This would be too much for their basic skill levels. Instead, they should learn how to use and work with tests.

Translation: 初学者不应该写测试。这对他们的基本技能水平来说太多了。相反，他们应该学习如何使用和使用测试。

Each programming course should center around exercises. I extend my exercises with unit tests and provide the students an environment which is already setup for running those tests.

Translation: 每门编程课程都应该以练习为中心。我用单元测试来扩展我的练习，并为学生提供一个已经设置好的环境来运行这些测试。

All the students have to do is write their code and then watch the lights of the testrunner turning from red to green. The resulting gamification is a nice side effect.

Translation: 学生们所要做的就是编写代码，然后看着测试人员的灯从红色变为绿色。由此产生的游戏化是一个很好的副作用。

For example: If the selected technology is Spring, I provide the exercises and tests within a Spring project. The students don’t need to know anything about Spring. All they need to know is the location of the exercises and the button to trigger the tests.

Translation: 例如：如果选择的技术是Spring，我将在Spring项目中提供练习和测试。学生们不需要知道任何关于春天的事情。他们只需要知道练习的位置和触发测试的按钮。

Additionally, students must know how to use a debugger and have a Read-Eval-Print Loop (REPL) handy. The ability to analyse code during runtime and to have a playground for small experiments is essential in TDD.

Translation: 此外，学生必须知道如何使用调试器，并且手边有Read-Eval-Print-Loop（REPL）。在TDD中，在运行时分析代码和拥有小实验场地的能力是必不可少的。

The main point is to ensure students don’t have to learn basic TDD behaviours after they’ve acquired core programming skills. Changing habits later in the students’ career will be much harder than learning those habits now. That’s why they should live and breath unit tests from the beginning.

Translation: 重点是确保学生在获得核心编程技能后不必学习基本的TDD行为。在学生职业生涯的后期改变习惯将比现在学习这些习惯困难得多。这就是为什么他们应该从一开始就进行生活和呼吸单元测试。

Later in their professional life, they should have an antipathy for projects without unit tests. They should intuitively see the absence of unit tests as anti-pattern.

Translation: 在以后的职业生涯中，他们应该会对没有单元测试的项目感到反感。他们应该直观地将单元测试的缺失视为反模式。

Fundamentals First

Translation: 基础知识第一

I hear very often that rookies should immediately start with a framework. This is like teaching people how to drive by placing them in a rally car and asking them to avoid oversteering. This simply ignores the fact that they still mistake the brake for the throttle.

Translation: 我经常听说新手应该立即从一个框架开始。这就像教人们如何驾驶，把他们放在拉力车上，并要求他们避免过度转向。这只是忽略了一个事实，即他们仍然将制动器误认为油门。

The same applies when we start students with a framework like Angular. Beginners need to understand the fundamentals of programming first. They need to be familiar with the basic elements and what it means to write code before they can use somebody else’s.

Translation: 当我们用Angular这样的框架开始学习时，同样的道理也适用。初学者首先需要了解编程的基本原理。在使用他人的代码之前，他们需要熟悉基本元素以及编写代码的含义。

The concept of a function, a variable, a condition, and a loop are completely alien to novices. These four elements build the foundations of programming. Everything a program is made of relies on them.

Translation: 函数、变量、条件和循环的概念对新手来说是完全陌生的。这四个要素奠定了编程的基础。程序的所有组成部分都依赖于它们。

Students are hearing these concepts for the very first time, but it is of the utmost importance that the students become proficient with them. If students do not master the fundamentals, everything that follows looks like magic and leads to confusion and frustration.

Translation: 学生们是第一次听到这些概念，但最重要的是让学生熟练掌握这些概念。如果学生不掌握基本原理，接下来的一切看起来都像魔术，会导致困惑和沮丧。

Teachers should spend more time on these fundamentals. But, sadly, many move on far too quickly. The problem is that some teachers struggle to put themselves into the role of a student. They have been programming for ages and have forgotten what types of problems a beginner has to deal with. It is quite similar to a professional rally driver. He can't imagine that somebody needs to think before braking. He just does it automatically.

Translation: 教师应该花更多的时间在这些基础知识上。但是，可悲的是，许多人的行动太快了。问题是，一些老师很难把自己放在学生的角色中。他们编程已经很长时间了，忘记了初学者必须处理哪些类型的问题。这与职业拉力赛车手非常相似。他无法想象有人在刹车前需要思考。他只是自动地做。

I design my exercises so that they are challenging but solvable in a reasonable amount of time by using a combination of the four main elements.

Translation: 我设计我的练习，使它们具有挑战性，但可以通过使用四个主要元素的组合在合理的时间内解决。

A good example is a converter for Roman and Arabic numbers. This challenge requires patience from the students. Once they successfully apply the four elements to solve the challenge, they also get a big boost in motivation.

Translation: 一个很好的例子是罗马数字和阿拉伯数字的转换器。这个挑战需要学生们的耐心。一旦他们成功地运用这四个要素来解决挑战，他们的动力也会大大增强。

Fundamentals are important. Don't move on until they are settled.

Translation: 基本面很重要。在问题解决之前不要继续前进。

Libraries and Frameworks

Translation: 库和框架

After students spend a lot of time coding, they must learn that most code already exists in the form of a library or a framework. This is more a mindset than a pattern.

Translation: 在学生花费大量时间进行编码之后，他们必须了解到大多数代码已经以库或框架的形式存在。这与其说是模式，不如说是一种心态。

As I have written before: Modern developers know and pick the right library. They don't spend hours writing a buggy version on their own.

Translation: 正如我之前所写的：现代开发人员知道并选择了正确的库。他们不会花几个小时自己写一个bug版本。

To make that mindset transition a success, the examples from the "fundamentals phase" should be solvable by using well-known libraries like Moment.js, Jackson, Lodash, or Apache Commons.

Translation: 为了使这种心态转变成功，"基础阶段"的例子应该可以通过使用Moment.js、Jackson、Lodash或Apache Commons等知名库来解决。

This way, students will immediately understand the value of libraries. They crunched their heads around those complicated problems. Now they discover that a library solves the exercise in no time.

Translation: 这样，学生将立即了解图书馆的价值。他们对那些复杂的问题绞尽脑汁。现在他们发现一个图书馆很快就能解决这个问题。

Similar to TDD, students should become suspicious when colleagues brag about their self-made state management library that makes Redux unnecessary.

Translation: 与TDD类似，当同事吹嘘他们自制的国家管理库使Redux变得不必要时，学生应该会产生怀疑。

When it comes to frameworks, students will have no problem understanding the importance once they understand the usefulness of libraries.

Translation: 当涉及到框架时，一旦学生理解了图书馆的有用性，他们就不会有问题地理解其重要性。

Depending on the course's timeframe, it may be hard to devote time to frameworks. But as I already pointed out, the most important aspect is shifting the mindset of the student away from programming everything from scratch to exploring and using libraries.

Translation: 根据课程的时间安排，可能很难将时间花在框架上。但正如我已经指出的，最重要的方面是将学生的心态从从头开始编程转变为探索和使用库。

I did not add tools to this pillar, since they are only of use to experienced developers. At this early stage, students do not need to learn how to integrate and configure tools.

Translation: 我没有在这个支柱中添加工具，因为它们只对有经验的开发人员有用。在这个早期阶段，学生不需要学习如何集成和配置工具。

Master & Apprentice

Translation: 硕士和学徒

In my early 20s I wanted to learn to play the piano. I did not want a teacher, and thought I could learn it by myself. Five years later, I consulted a professional tutor. Well, what can I say? I've learned more in 1 month than during the five years before.

Translation: 在我20岁出头的时候，我想学弹钢琴。我不想要老师，我想我可以自学。五年后，我咨询了一位专业导师。好吧，我能说什么呢？我在一个月内学到了比五年前更多的东西。

My piano teacher pointed out errors in my playing I couldn't hear and made me aware of interpretational things I never would have imagined. After all, she instilled in me the mindset for music and art, both of which were out of reach for me as a technical person.

Translation: 我的钢琴老师指出了我演奏中听不见的错误，并让我意识到了我从未想过的解释性事情。毕竟，她给我灌输了音乐和艺术的心态，这两种心态对我这个技术人员来说都是遥不可及的。

It is the same in programming. If somebody has no experience in programming, then self-study can be a bad idea. Although there are many success stories, I question the efficiency of doing it alone.

Translation: 在编程中也是如此。如果有人没有编程经验，那么自学可能是个坏主意。尽管有很多成功的故事，但我质疑单独做这件事的效率。

Instead, there should be a “master & apprentice” relationship. In the beginning, the master gives rules the apprentice must follow — blindly! The master may explain the rules, but usually the reasoning is beyond the apprentice’s understanding.

Translation: 相反，应该有一种“师徒关系”。一开始，师傅给学徒制定了必须遵守的规则--盲目！师傅可以解释规则，但通常推理超出了学徒的理解范围。

These internalised rules form a kind of safety net. If one gets lost, one always has some safe ground to return to.

Translation: 这些内化的规则形成了一种安全网。如果一个人迷路了，总有一些安全的地方可以回去。

Teaching should not be a monologue. The master has to deal with each student individually. He should check how the students work, give advice, and adapt the speed of the course to their progress.

Translation: 教学不应该是独白。大师必须单独对待每个学生。他应该检查学生的工作方式，给出建议，并根据他们的进步调整课程的速度。

Once the apprentices reach a certain level of mastery, they should be encouraged to explore new territory. The master evolves into a mentor who shares “wisdom” and is open for discussions.

Translation: 一旦学徒达到一定的熟练程度，就应该鼓励他们探索新的领域。大师演变成一个分享“智慧”并乐于讨论的导师。

Challenge and Motivation

Translation: 挑战与动力

“Let’s create a Facebook clone!” This doesn’t come from a CEO backed by a horde of senior software developers and a multi-million euro budget. It is an exercise from an introductory course for programmers. Such an undertaking is virtually impossible. Even worse, students are put into wonderland and deluded into believing they have skills that are truly beyond their reach.

Translation: “让我们创建一个Facebook克隆！”这并不是来自一位由一群资深软件开发人员和数百万欧元预算支持的首席执行官。这是程序员入门课程中的一个练习。这样的事业几乎是不可能的。更糟糕的是，学生们被带入了仙境，被欺骗，相信自己拥有真正遥不可及的技能。

No doubt the teacher is aware of that, but creates such exercises for motivational reasons.

Translation: 毫无疑问，老师意识到了这一点，但出于动机的原因，他创建了这样的练习。

The main goal of an exercise is not to entertain. It should be created around a particular technique and should help the students understand that technique.

Translation: 练习的主要目的不是娱乐。它应该围绕一种特定的技术来创建，并帮助学生理解这种技术。

Motivation is good, but not at the sacrifice of content. Programming is not easy. If the students don’t have an intrinsic motivation, coding might not be the way to go.

Translation: 动机是好的，但不能以牺牲内容为代价。编程并不容易。如果学生们没有内在的动机，那么编码可能就不是一条出路。

Newbies should experience what it means to be a professional developer. They should know what awaits them before they invest lots of hours.

Translation: 新手应该体验一下成为一名专业开发人员意味着什么。在投入大量时间之前，他们应该知道等待他们的是什么。

For example, many business applications center around complex forms and grids. Creating these is an important skill that exercises can impart. Building an application similar to Facebook might not be the best lesson for students to learn right away.

Translation: 例如，许多业务应用程序都以复杂的表单和网格为中心。创造这些是练习可以传授的一项重要技能。构建一个类似于Facebook的应用程序可能不是学生立即学习的最佳课程。

Similarly, a non-programmer might be surprised at how few code lines a developer writes per day. There are even times where we remove code or achieve nothing.

Translation: 同样，非程序员可能会惊讶于开发人员每天编写的代码行如此之少。甚至有时我们删除代码或一无所获。

Why? Because things go wrong all the time. We spend endless hours fixing some extremely strange bugs that turn out to be a simple typo. Some tool might not be working just because a library got a minor version upgrade. Or the system crashes because somebody forgot to add a file to git. The list can go on and on.

Translation: 为什么？因为事情总是出问题。我们花了无数的时间来修复一些极其奇怪的错误，这些错误原来只是一个简单的拼写错误。某些工具可能无法正常工作，因为库进行了小版本升级。或者由于有人忘记向git添加文件而导致系统崩溃。这个名单可以一直列下去。

Students should enjoy these experiences. An exercise targeting an unknown library under time pressure might be exactly the right thing. ;)

Translation: 学生们应该享受这些经历。在时间压力下针对未知图书馆进行练习可能正是正确的做法。）

The sun isn't always shining in real life. Beginners should be well-prepared for the reality of programming.

Translation: 在现实生活中，阳光并不总是灿烂的。初学者应该为编程的现实做好充分的准备。

Final Advice

Translation: 最后建议

Last but not least: One cannot become a professional programmer in two weeks, two months or even a year. It takes time and patience.

Translation: 最后但同样重要的是：一个人不可能在两周、两个月甚至一年内成为一名专业程序员。这需要时间和耐心。

Trainers should not rush or make false promises. They should focus on whether students understand the concepts and not move on too fast.

Translation: 培训师不应操之过急或做出虚假承诺。他们应该关注学生是否理解这些概念，不要走得太快。

Originally published at www.rainerhahnekamp.com on June 10, 2018.

Translation: 最初于2018年6月10日发布于www.rainerahnekamp.com。